IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 13, No. 1, March 2024, pp. 358∼367
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i1.pp358-367

Detecting cyberbullying text using the approaches with machine learning models for the low-resource Bengali language

Md. Nesarul Hoque[1,2], Md. Hanif Seddiqui[1]
[1]Department of Computer Science and Engineering, Faculty of Engineering, University of Chittagong, Chittagong, Bangladesh
[2]Department of Computer Science and Engineering, Engineering Faculty, Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Gopalganj, Bangladesh

Article Info

Article history:

Received Jan 30, 2023
Revised Apr 21, 2023
Accepted May 7, 2023

Keywords:

Bengali
Cyberbullying
Feature selection
Low-resource language
Machine learning
Pre-processing

ABSTRACT

The rising usage of social media sites and the advances in communication technologies have led to a considerable increase in cyberbullying events. Here, people are intimidated, harassed, and humiliated via digital messaging. To identify cyberbullying texts, several research have been undertaken in English and other languages with abundant resources, but relatively few studies have been conducted in low-resource languages like Bengali. This research focuses on Bengali text to find cyberbullying material by experimenting with pre-processing, feature selection, and three types of machine learning (ML) models: classical ML, deep learning (DL), and transformer learning. In classical ML, four models, support vector machine (SVM), multinomial Naive Bayes (MNB), random forest (RF), and logistic regression (LR) are used. In DL, three models, long short term memory (LSTM), Bidirectional LSTM, and convolutional neural network with bidirectional LSTM (CNN-BiLSTM) are employed. As the transformer-based pre-trained model, bidirectional encoder representations from transformers (BERT) is utilized. Using our proposed pre-processing tasks, the MNB-based approach achieves the best accuracy of 78.816% among the other classical ML models, the LSTM-based approach gains the highest result of 77.804% accuracy among the DL models, and the BERT-based approach outperforms both with 80.165% accuracy.

Corresponding Author:

Md. Nesarul Hoque
Department of Computer Science and Engineering, Faculty of Engineering, University of Chittagong
Chittagong-4331, Bangladesh
Email: mnhshisir@gmail.com

1. INTRODUCTION

Social communication platforms like Facebook, Instagram, YouTube, and others are now heavily used for disseminating personal feelings and opinions. According to a report in 2021, about 2.9 billion users actively interact with Facebook each month, and 2.4 billion and 1.4 billion people, respectively, prefer YouTube for sharing videos and Instagram for photos. During this sharing, a person is often threatened or embarrassed by sending unpleasant messages through electronic technology. Thus, the term cyberbullying comes, and it is growing throughout the world. Sometimes, it results in extreme emotional and physical torment and even self-destruction [1].

According to the report of NapoleonCat, in 2021, almost 47 million Bangladeshi users interact with Facebook, and about 68.8% of these users are male. As per a survey of the Global Digital Statshot, in terms of the highest number of active users, Dhaka, the capital of Bangladesh, ranked second among all cities globally in 2017. The Daily Star, a leading newspaper in this country, reported in 2020 that there is a high incidence of cyberbullying in this region, with 80% of the victims between the ages of 14 and 22 being females.

Although, cyberbullying is the rising issue in Bangladesh, few research articles have been published to detect Bengali bully text. However, identifying the cyberbullying class is not the easy task, since extracting the intention and the context of a particular text is still challenging jobs for the researchers. Furthermore, very rare of the study focus on detail analysis of text pre-processing and feature selection levels from three aspects of machine learning (ML) models: classical ML, deep learning (DL), and transformer learning [2]. Moreover, most of these studies do not consider cross-validation technique to develop more generic detection model. In addition, we have found very less analysis of hyper-parameter tuning of the DL models. In this research, we want to handle these issues in order to build a strong cyberbullying text detection system for the Bengali language.

Several pieces of research have been undertaken to investigate bullying materials in the Bengali language. Ahmed *et al.* [3] successfully identified bullying texts in the social media texts. Here, the authors used a unique feature, the pointwise mutual information-semantic orientation (PMI-SO) score, besides the other features like number of likes, and cyberbullying words. Then, they applied the extreme gradient boosting (XGBoost) classifier and obtained a 93% accurate result. However, they did not apply cross-validation or hyper-parameter tuning, and they did not try DL or transformer-based models in this work. Ahmed *et al.* [4] made three cyberbullying detection systems for three different types of Bengali datasets: single-coded Bengali, Romanized Bengali, and a mix of single-coded and Romanized Bengali. This is the unique contribution in the case of cyberbullying detection. For single-coded Bengali, they proposed the convolutional neural network (CNN) classifier (with an accuracy of 84%); however, for the other two dataset types, multinomial Naive Bayes (MNB) showed superior performance: 84% for the Romanized Bengali and 80% for the mixed dataset. They did not apply transformer-based pre-trained models and did not use cross-validation or hyper-parameter optimization techniques in this research. Das *et al.* [5] worked on a hateful dataset and handled both binary and multi-class classification problems. Here, they created the "Bangla Emot" module for processing emoji and emoticons and provided an attention-based decoder model that demonstrated the best performance with an accuracy of 88% in binary classification and 77% in multi-classification. Cross-validation and hyper-parameter tuning were not taken into account. Romim *et al.* [6] conducted an experiment on hateful text. For extracting the features, they applied fastText, BengFastText, and word2vec word embedding techniques. The authors proposed the support vector machine (SVM) model that successfully achieved the highest accuracy of 87.5% and the highest F1-score of 0.911. However, hyper-parameter tuning and cross-validation were not considered in this work. Also, they did not experiment on the transformer-based models to detect the hateful text. Islam *et al.* [7] utilized SVM model with term frequency-inverse document frequency (TF-IDF) feature extraction technique and achieved 88% accuracy on an abusive text dataset. Authors did not perform cross-validation and hyper-parameter optimization. Moreover, this research did not consider DL and transformer learning methods to identify the abusive text. Kumar *et al.* [8] employed two shared task datasets, "HASOC" [9] and "TRAC-2" [10], to identify offensive and aggressive text in Bengali, Hindi, and English languages. The principal success of this research is the authors achieved the best result (0.93 F1-score) using a mix of character trigram and word unigram with a SVM classifier for Bengali text. Although they applied 5-fold cross-validation, hyper-parameter optimization was skipped. Chakraborty and Seddiqui [11] investigated abusive and threatening texts and developed a detection system through the linear SVM model with greater accuracy (78%), which is the main success of this study. The authors did not use the cross-validation technique for generalizing the model and did not tune the hyper-parameters during the model execution. Furthermore, they did not choose transformer learning algorithms during model implementation.

Our goal in this study was to address every concern of the previous studies in this area. With this in mind, we conduct experiments on three levels: pre-processing tasks, feature selection tasks, and implementation of cutting-edge ML models. Ultimately, we contribute the following to the establishment of a very effective cyberbullying detection system:

– Pre-processing level: incorporate some additional pre-processing tasks such as removing thin-space Unicode characters, replacing emoticons and emojis by the Bengali words, with the general pre-processing tasks.

– Feature selection level: features are extracted for the classical ML, DL, and transformer leaning classifiers separately.
– Cross-validation technique: perform 10-fold cross-validation approach for classical ML models for more reliable detection system.
– Hyper-parameter tuning: perform hyper-parameter tuning using “Bayesian optimization” algorithm during the implementation of DL models.
– Selection of best models: propose the better detection approaches from each three type of ML model.

2. METHOD

For developing a strong detection system, at first we collect the dataset. Then, we apply various text pre-processing tasks. After that, we extract the features from the pre-processed data. Finally, we apply different types of classifier models with 10-fold cross-validation and hyper-parameters tuning using Bayesian optimization technique. The overall working process is depicted in Figure 1. The following sub-sections explain all the processes step-by-step.

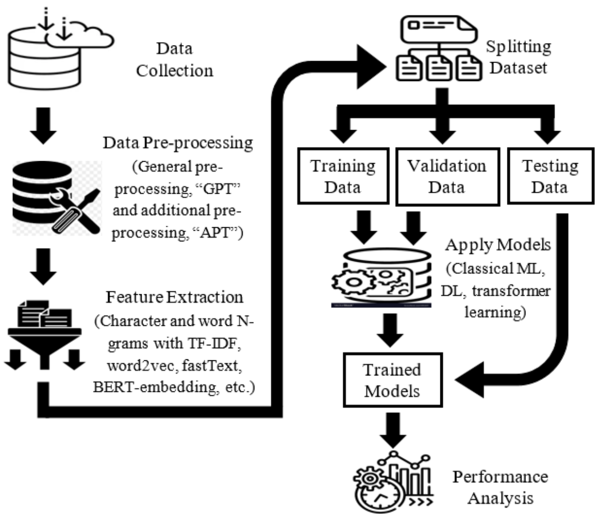
Data Collection
Data Pre-processing (General pre-processing, “GPT” and additional pre-processing, “APT”)
Feature Extraction (Character and word N-grams with TF-IDF, word2vec, fastText, BERT-embedding, etc.)
Splitting Dataset
Training Data
Validation Data
Testing Data
Apply Models (Classical ML, DL, transformer learning)
Trained Models
Performance Analysis

Figure 1. Overall working process

2.1. Dataset description

In this investigation, the same dataset was utilized as in [11]. This dataset contains 5,644 code-mixing Bengali-English comments and posts which are collected from various popular types of Facebook pages like celebrities, newspapers, and entertainments. Here, the almost 50% data are tagged as bullying and the remaining are non-bullying. Table 1 presents some statistical information about the dataset including, number of entries (quantity), minimum (minWords), maximum (maxWords), and average (avgWords) number of words in a text.

Table 1. Statistical information of the dataset

Class	quantity	minWords	maxWords	avgWords
Bullying	2,718	1	201	19
Non-bullying	2,926	2	247	23
Bullying and Non-bullying	5,644	1	247	21

2.2. Data pre-processing

The collected data convey unwanted and noisy contents such as thin-space Unicode characters, links, and punctuation. Here, we have divided entire pre-processing tasks into two groups: general pre-processing tasks named “GPT” and additional pre-processing tasks called “APT”. In the “GPT”, we remove HTML tags, links, URLs, special characters, punctuation, digits, and stop-words and apply tokenization, lower casing (for English words), and stemming [12] operations. On the other hand, the “APT” contains the following four pre-processing tasks:

– Removing thin-space unicode character: our dataset contains “ZERO WIDTH NON-JOINER” (U+200C) Unicode character which creates garbage tokens when we use regular expression during tokenization. We have discarded this unicode character from the dataset.
– Fixing space error with delimiters and sentence-ending punctuation: in this dataset, we have found many space related errors before and after delimiters like comma (“,”), semi-colon (“;”), and sentence-ending punctuation like full-stop (“.”) or (“।”), question mark (“?”). These types of errors are corrected that reduces the number of unwanted tokens.
– Replacing emoticons (ASCII characters) with Bengali words: we have developed an emoticon dictionary which contains about 400 usable emoticons and their equivalent Bengali meaning. Here, we have grouped the similar types of emoticons with the same Bengali word. E.g., the emoticons, “:)”, “:=)”, “:|)”, convey the same Bengali meaning, “খুশি” (smile). Entire emoticons of the dataset are substituted with corresponding Bengali words. This replace operation may help for better understanding of the emotion [13] of a particular text.
– Replacing emoji (unicode characters) with Bengali words: like the emoticons, we have replaced every emoji by the equivalent Bengali words. Here, we also make an emoji dictionary of about 280 mostly used emoji with their Bengali meaning and group the similar types of emoji with same the Bengali word.

To the best of our knowledge, the above four pre-processing tasks are slightly differed from those of existing studies. We experiment on “APT” in section 3. Finally, we observe the impact of these pre-processing tasks in the detection system.

2.3. Feature selection

A feature selection operation has been introduced to extract the significant features that have a substantial impact on cyberbullying text detection. We have addressed feature selection technique from three different perspectives: features for traditional ML models, features for DL models, and features for transformer-based pre-trained model. For traditional ML models, we have utilized N-grams at the character and word level and assigned TF-IDF values for each feature. After conducting some empirical research, we have determined that N should be between 3 and 5 for character N-grams and between 1 and 2 for word N-grams. For DL models, we use three word embedding models: word2vec, fastText, and global vector (GloVe) which deal with syntactic and semantic word-similarity and word-analogy of a text. As a transformer-based ML technique, we have utilized a cutting-edge bidirectional encoder representations from transformers (BERT) pre-trained model. During feature extraction, BERT employs its own tokenizer, the BERT tokenizer, and turns each word into three embedding vectors: token, segment, and position. By adding these three vectors, the final embedding of a particular word is computed [14].

2.4. Model classifiers

This research experiments on eight ML models such as SVM, MNB, random forest (RF), logistic regression (LR), long short term memory (LSTM), bidirectional LSTM (BiLSTM), CNN-BiLSTM, and BERT, which give better outputs in several studies for classifying a text for the Bengali language. In the case of classical ML models, the SVM model outperformed in [3], [7], [8], [11], whereas, the MNB model showed better results in [4], [15]. However, the RF and the LR presented the higher accuracy in [16] and [17], respectively. On the other side, LSTM, BiLSTM, and CNN-BiLSTM presented the best performance in [18], [19], respectively in the DL models. At very recent time, BERT plays outstanding result in text classification task [20]. These eight models are illustrated below:

Support vector machine (SVM): in SVM, a hyper-plane space is generated by maximizing the gap between the margins of two classes (E.g., positive and negative) of support vectors. This marginal distance yields a more generalized version of the model. In addition, SVM handles errors by employing a regularization technique based on the soft margin hyper-plane over incorrectly categorized data points [21].

Multinomial Naive Bayes (MNB): the main working principle of MNB is based on the Bayes theorem [22]. Through this theorem and the formula of prior probability and the conditional probability, each class variable and each feature variable are computed during the training process. After that, these prior and conditional probabilities are applied on the test dataset. Then, based on the features of the test data, MNB assigns the probability for the classes of each data point and select a class with higher probability. The probability, $P(C|F)$ is calculated using the Bayes theorem as (1).

$$P(C|F) = P(F|C) * P(C)/P(F) \tag{1}$$

Where, C is a possible class variable and F represents the unique feature.

Random forest (RF): the RF consist of a number of decision trees which are computed separately [23]. Here, the information gain is achieved for each tree through the “Gini Impurity” formula as follows:

$$Gini = 1 - \sum_{n=1}^{C}(P_i)^2 \tag{2}$$

where, C is the total number of possible class and P_i is the probability of i^{th} class.

Logistic regression (LR): the working principle of this model is based on the two functions: sigmoid (3) and cost (4), given below:

$$h_\theta(x) = \frac{1}{1 + e^{-\beta^T x}} \tag{3}$$

$$C(\theta) = \frac{1}{k}\sum_{i=1}^{k} c(h_\theta(x_i), y_i) \tag{4}$$

$$c(h_\theta(x), y) = \begin{cases} -\log(1 - h_\theta(x)), & \text{if } y = 0 \\ -\log(h_\theta(x)), & \text{if } y = 1 \end{cases}$$

where, β^T indicates the transpose of regression coefficient matrix (β), k represents the total amount of training observations, $h_\theta(x_i)$ is the hypothesis function of the i^{th} training observation, and y_i presents the actual output of i^{th} training sample.

Long short term memory (LSTM): this model is developed to handle the problem of long-term dependency for the sequential and the time series data [24]. Here, four interactive layers: memory cell unit, forget gate unit, input gate unit, and output gate unit are working together in each LSTM cell state. Information are added or forgot in the memory cell state. Forget gate unit decides which information from the previous cell state. In the input gate unit, new significant information are added into the memory cell state. Lastly, the output gate unit decides which information are passed into the next LSTM cell state.

Bidirectional LSTM (BiLSTM): the LSTM model struggles to predict a word which more depend on the future context word. To solve this issue, the concept of BiLSTM where one LSTM operates into forward direction and another one flows into backward directions [25]. Here, the output of a particular BiLSTM cell state is computed by combining the output of both LSTM cells for the same time-stamp.

Convolutional neural network with bidirectional LSTM (CNN-BiLSTM): the CNN-BiLSTM is a hybrid model, where CNN and BiLSTM models are executed sequentially. CNN operates with three fundamental layers: convolution, pooling, and fully connected. In the convolution layer, element by element matrix multiplication operation (convolution operation) is performed between the input data and the kernel (feature detector) and produce a feature map for each kernel. The dimension of this feature map is reduced in the pooling layer. Finally the reduced feature map is passed through the fully connected layer which performs like the standard artificial neural network (ANN) architecture. In the CNN-BiLSTM model, the BiLSTM part is added just before the fully connected layer. For the text processing task, CNN works on local dependency of the contiguous words of a sentence by extracting character level features, while BiLSTM deals with overall dependency for a particular feature of an entire text [26].

Bidirectional encoder representations from transformers (BERT): BERT shows an utmost performance for context-specific tasks including language inference, question answering, among the other pre-trained models like embeddings from language models (ELMo), and OpenAI generative pre-training (GPT). It works with two separate frameworks: pre-training framework and fine-tuning framework. In the pre-train framework, BERT obtains the contextual understand about the specified languages using two unsupervised tasks: mask language model (MLM) and next sentence prediction (NSP). In contrast, the fine-tuning framework is applied to several downstream tasks linked to natural language processing (NLP), such as sentiment classification, sentence prediction, and named entity recognition, by simple adjustment of the task-specific architecture [14].

2.5. Experimental set-up

We have explained the experimental procedures and model configuration from three points of view: classical ML models, DL models, and transformer-based pre-trained models. We examine the potential combinations of pre-processing tasks, feature selection tasks, and classifier models. In addition, we configure the hyper-parameters during the implementation of the detection system.

2.5.1. Experimental set-up for classical ML models

For evaluating the influence of the pre-processing task "additional pre-processing tasks" (APT) as shown in section 2.2., we experiment from two point of views: "general pre-processing tasks" (GPT) with "APT" and "GPT" without "APT". In feature selection stage, three approaches are settled by empirical experiments: character N-grams (3 to 5) with TF-IDF, word N-grams (1 to 2) with TF-IDF, and a combined character-word N-grams. In this investigation, four ML classifiers, including SVM, MNB, RF, and LR, are utilized. Consequently, each classifier provides six potential outputs. During the experiment, the dataset is separated into two parts: training data and testing data, with respective proportions of 80% and 20%. In order to obtain a more reliable model, a 10-fold cross-validation procedure is employed [27]. Using sklearn Python libraries, we have tried to use default values of hyper-parameters to implement each model. However, slight changes of the hyper-parameter values are taken into account through the empirical experimentation such as kernel is "linear" and gamma is "auto" for SVM model, and alpha is 0.25 for MNB model.

2.5.2. Experimental set-up for DL models

During the implementation of DL models, same two variations of pre-processing tasks are chosen like classical ML models. However, three different feature selection techniques such as word2vec, fastText, and GloVe are applied for three DL models, such as LSTM, BiLSTM, and CNN-BiLSTM. Therefore, each DL model shows six possible results. Here, we use the scikit-learn python library and KerasTuner framework to implement the models by tuning the models' parameters. We have experimented on thirteen parameters which value ranges are specified in Table 2 via the empirical analysis. We implement the LSTM model with two LSTM layers, the BiLSTM model with two BiLSTM layers, and the CNN-BiLSTM model with one BiLSTM layer. Here, the dataset is split into three parts: training data, validation data, and testing data with a ratio of 70%, 15%, and 15%.

Table 2. Hyper-parameter values during the implementation of DL models

Parameter	Data Type	Description	Value
The number of convolution layers	Integer	Number of convolution layers in CNN.	Min value=2 and Max value=3
Filters	Integer	The number of output filters in each convolution.	Min value=32, Max value=768, and Step value=32
Kernel size	Integer	Indicating the length of the convolution window.	Min value=3, and Max value=5
Pooling types	String	Max pooling and average pooling, used to reduce the dimensionality of the feature map.	value=['max pooling', 'average pooling']
LSTM units	Integer	Number of LSTM output units.	Min value=32, Max value=768, and Step value=32
Number of hidden layers	Integer	The number of hidden layers in the fully connected network.	Min value=1 and Max value=3
Hidden units	Integer	The number of neurons of each hidden layer.	Min value=32, Max value=512, and Step value=32
Activation function	String	The function defines the output value of each neuron.	Value=['relu', 'leaky_relu', 'elu', 'swish']
Kernel initializer	String	A procedure to assign a set of small random values of a neural network at the very early stage.	Value= ['glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform']
Dropout rate	Float	Fraction of the number of neurons to drop.	Min value=0.2, Max value=0.5, and Step value=0.1
Learning rate (Adam)	Float	It controls the rate at which a loss function approaches the point where the curves converge.	Value=[1e-1, 1e-2, 1e-3, 1e-4]
Batch size	Integer	The number of samples passing in each iteration during training the model.	Min value=16, Max value=256, and Step value=16
Epochs	Integer	The number of times the entire training data is used to train the model.	Min value=5, Max value=30

2.5.3. Experimental set-up for transformer-based pre-trained model

As a transformer learning technique, we have used BERT-base multilingual pre-trained model consist of 12 hidden layers, hidden output dimensions of 768, and 12 self-attention heads. Through the empirical study, we have selected four categories of pre-processing tasks, including: "GPT" with "APT", "GPT" with "APT", but not applying stop-word removal and stemming operations, "GPT" without "APT", and "GPT" without "APT", stop-word removal, and stemming operations. Since, in the BERT fine-tuning, the feature vectors come through the BERT's own tokenizer and embedding technique. Therefore, we get total four possible outputs from a BERT model.

3. RESULTS AND DISCUSSION

We illustrate the results of each ML-based approach and their performance analysis. The performance score is determined using assessment criteria, including precision, recall, F1-score, and accuracy metrics. At first, we analyze the results from three angles: approaches with classical ML models, approaches with DL models, and with a transformer-based pre-trained model. Finally, we summarize the overall results.

3.1. Results of approaches with classical ML models

The performance scores of the combination of pre-processing, feature selection, and classical ML models (MNB, LR, SVM, and RF) are pointed in Table 3. The MNB-based approach performs the highest F1-score of 0.745 and the highest accuracy of 78.815% which crosses the performance of the existing work [11]. In this approach, the "APT" is included into the "GPT" in the pre-processing stage, and the combined N-grams is utilized in feature selection stage. Here, both the "APT" and the combined N-grams tasks help in calculating the proper feature probability by (1), which distinguishes the bully and the non-bully type text effectively. The MNB-based approach also gives better output for character N-grams of 76.759% accuracy. This approach obtains the top recall score too by 72.566%, however, word N-grams is taken instead of combined N-grams. In the case of precision, the LR-based approach gives maximum output of 82.499%. This approach shows comparatively better accuracy than the other two classical ML models, SVM and RF. Here, the hypothesis function (3) of the LR model effectively works on the training observation as well as test data for detecting bully text.

Table 3. Performance evaluation of the methods with classical ML classifiers

Approach	Precision	Recall	F1-score	Accuracy
"GPT" with "APT" + Character N-grams + SVM	79.117%	67.858%	0.716	76.582%
"GPT" with "APT" + Word N-grams + SVM	79.165%	64.290%	0.697	75.341%
"GPT" with "APT" + Combined N-grams + SVM	76.046%	69.844%	0.716	75.607%
"GPT" without "APT" + Character N-grams + SVM	79.027%	67.197%	0.712	76.387%
"GPT" without "APT" + Word N-grams + SVM	79.308%	63.443%	0.693	75.129%
"GPT" without "APT" + Combined N-grams + SVM	76.310%	69.366%	0.715	75.696%
"GPT" with "APT" + Character N-grams + MNB	81.703%	69.477%	0.737	78.620%
"GPT" with "APT" + Word N-grams + MNB	76.696%	**72.566%**	0.736	76.759%
"GPT" with "APT" + Combined N-grams + MNB	80.557%	71.611%	**0.745**	**78.815%**
"GPT" without "APT" + Character N-grams + MNB	81.899%	69.036%	0.736	78.620%
"GPT" without "APT" + Word N-grams + MNB	76.106%	71.867%	0.730	76.245%
"GPT" without "APT" + Combined N-grams + MNB	80.389%	71.170%	0.742	78.656%
"GPT" with "APT" + Character N-grams + RF	80.541%	65.136%	0.705	76.370%
"GPT" with "APT" + Word N-grams + RF	79.040%	61.603%	0.684	74.403%
"GPT" with "APT" + Combined N-grams + RF	81.025%	63.995%	0.699	76.264%
"GPT" without "APT" + Character N-grams + RF	80.677%	65.026%	0.707	76.600%
"GPT" without "APT" + Word N-grams + RF	76.621%	63.224%	0.683	73.801%
"GPT" without "APT" + Combined N-grams + RF	81.332%	64.952%	0.709	76.866%
"GPT" with "APT" + Character N-grams + LR	81.707%	66.387%	0.717	77.397%
"GPT" with "APT" + Word N-grams + LR	81.923%	59.692%	0.676	74.952%
"GPT" with "APT" + Combined N-grams + LR	81.209%	67.086%	0.721	77.397%
"GPT" without "APT" + Character N-grams + LR	81.869%	66.535%	0.721	77.539%
"GPT" without "APT" + Word N-grams + LR	**82.499%**	58.772%	0.673	74.863%
"GPT" without "APT" + Combined N-grams + LR	81.523%	67.050%	0.722	77.539%
SVM-based approach [11]	-	-	-	78.000%

3.2. Results of approaches with DL models

The evaluation scores of various combination of pre-processing tasks, feature selection methods, and DL models (LSTM, BiLSTM, and CNN-BiLSTM) are mentioned in Table 4. The LSTM-based approach achieves the highest recall, F1-score, and accuracy values of 75.000%, 0.765, and 77.804%, respectively which outperforms than the earlier work [11] on the same dataset. This approach uses ”APT” with ”GPT” pre-processing task and utilizes the fastText word embedding technique for selecting feature vectors. Since, the average number of words represents a small figure of 21 as shown in Table 1, the unidirectional LSTM model performs better result than the other two DL models (BiLSTM and CNN-BiLSTM). On the other hand, CNN-BiLSTM-based approach presents the top precision score of 85.161%. This approach also use the same pre-processing task and feature selection technique like the LSTM-based approach. Furthermore, this approach also gives the slightly better performance than [11] in terms of accuracy (77.568%). Another notifying point is that, the fastText embedding method performs much better outputs compare to the word2vec and the GloVe embedding methods in every cases. The main reason is the fastText method uses character N-grams approach and generates sub-word level features which distinguishes other two embedding methods. Using this mechanism, the fastText also handles unknown word dictionary of the dataset [28], [29].

Table 4. Performance evaluation of the methods with DL classifiers

Approach	Precision	Recall	F1-score	Accuracy
”GPT” with ”APT” + word2vec + LSTM	74.702%	61.520%	0.675	71.429%
”GPT” with ”APT” + fastText + LSTM	78.061%	**75.000%**	**0.765**	**77.804%**
”GPT” with ”APT” + GloVe + LSTM	75.313%	59.069%	0.662	70.956%
”GPT” without ”APT” + word2vec + LSTM	51.843%	55.147%	0.534	53.719%
”GPT” without ”APT” + fastText + LSTM	80.508%	69.853%	0.748	77.332%
”GPT” without ”APT” + GloVe + LSTM	70.833%	70.833%	0.708	71.901%
”GPT” with ”APT” + word2vec + BiLSTM	72.702%	63.971%	0.681	71.074%
”GPT” with ”APT” + fastText + BiLSTM	78.495%	71.569%	0.749	76.860%
”GPT” with ”APT” + GloVe + BiLSTM	73.158%	68.137%	0.706	72.609%
”GPT” without ”APT” + word2vec + BiLSTM	76.052%	57.598%	0.656	70.838%
”GPT” without ”APT” + fastText + BiLSTM	81.143%	69.608%	0.749	77.568%
”GPT” without ”APT” + GloVe + BiLSTM	70.051%	67.647%	0.688	70.484%
”GPT” with ”APT” + word2vec + CNN-BiLSTM	77.419%	58.824%	0.669	71.901%
”GPT” with ”APT” + fastText + CNN-BiLSTM	**85.161%**	64.706%	0.735	77.568%
”GPT” with ”APT” + GloVe + CNN-BiLSTM	69.576%	68.382%	0.690	70.366%
”GPT” without ”APT” + word2vec + CNN-BiLSTM	74.566%	63.235%	0.684	71.901%
”GPT” without ”APT” + fastText + CNN-BiLSTM	79.348%	71.569%	0.753	77.332%
”GPT” without ”APT” + GloVe + CNN-BiLSTM	74.167%	65.441%	0.695	72.373%
CNN-LSTM-based approach [11]	-	-	-	77.500%

3.3. Results of approaches with Transformer-based pre-trained model

In this experiment, we have implemented the BERT pre-trained model with it’s own tokenizer and embedding method for four combination of pre-processing tasks between “APT” and stop-word removal and stemming as shown in Table 5. Here, only the recall reaches on the top when “APT” is not considered with “GPT”, however other three cases: precision, F1-score, and accuracy, BERT-based approach obtains the maximum values of 84.682%, 0.777, and 80.165%, respectively, by incorporating “APT” to other pre-processing tasks except not performing stop-word removal and stemming operations. Since, the BERT obtains a contextual understanding about the languages by training from a very large dataset, thus, stop-words removing and stemming tasks may deteriorate comprehensive understanding about a text. For that reason, if we do not consider these two pre-processing tasks, the result shows better output values.

Table 5. Performance evaluation of the methods with transformer model

Approach	Precision	Recall	F1-score	Accuracy
“GPT” with “APT” + BERT-embedding + BERT-base-multilingual-uncased	77.778%	73.775%	0.757	77.214%
“GPT” with “APT” but not perform stop-word removal and stemming + BERT-embedding + BERT-base-multilingual-uncased	**84.682%**	71.814%	**0.777**	**80.165%**
“GPT” without “APT” + BERT-embedding + BERT-base-multilingual-uncased	74.146%	**74.510%**	0.743	75.207%
“GPT” without “APT”, stop-word removal, and stemming + BERT-embedding + BERT-base-multilingual-uncased	83.380%	72.549%	0.776	79.811%

3.4. Overall result analysis

We have extracted the best outputs in terms of F1-score and accuracy from each three types of ML models where the BERT-based approach gives outstanding results to detect the cyberbullying text as shown in Table 6. We observe that our proposed pre-processing task, "APT", improves the overall performance of the detection system. Furthermore, in the feature selection stage, combining character and word N-grams with TF-IDF scoring, the fastText embedding, and the BERT-embedding techniques are effective in classical ML, DL, and BERT-based transformer models, respectively.

Table 6. Proposed approaches regarding F1-scores and accuracy values

Proposed Approach	F1-score	Accuracy
"GPT" with "APT" + Combined N-grams + MNB	0.745	78.815%
"GPT" with "APT" + fastText + LSTM	0.765	77.804%
"GPT" with "APT" but not perform stop-word removal and stemming + BERT-embedding + BERT-base-multilingual-uncased	**0.777**	**80.165%**

4. CONCLUSION

Within this study, we have experimented with pre-processing tasks, feature selection tasks, and selection of classifier models from three points of view: classical ML, DL, and transformer learning, and proposed a concise decision in every aspect. Here, "APT" pre-processing task is introduced besides the general pre-processing tasks. Then, we have shown the combined N-grams (character and word level) performs well for the classical ML algorithms, while, fastText gives better results than the other two embedding techniques such as word2vec and GloVe for the DL algorithms. Finally, using "APT", BERT outperforms with an utmost accuracy of 80.165%. In future research, we will implement our proposed approaches to the other related datasets. Furthermore, we will analyze other new pre-processing tasks and will extract new features. Finally, we will experiment with various ML approaches (classical ML, DL, and transformer learning) separately and in a combined form.

ACKNOWLEDGEMENTS

The Information and Communication Technology (ICT) division (Government of the People's Republic of Bangladesh) provides a fellowship whose code number is 1280101-120008431-3821117 for the 2022-2023 fiscal year. This fellowship supports our entire research work.

REFERENCES

[1] D. L. Hoff and S. N. Mitchell, "Cyberbullying: causes, effects, and remedies," *Journal of Educational Administration*, vol. 47, no. 5, pp. 652–665, Aug. 2009, doi: 10.1108/09578230910981107.

[2] M. N. Hoque, P. Chakraborty, and M. H. Seddiqui, "The challenges and approaches during the detection of cyberbullying text for low-resource language: A literature review," *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 17, no. 2, pp. 192–214, Apr. 2023, doi: 10.37936/ecti-cit.2023172.248039.

[3] M. Ahmed, M. Rahman, S. Nur, A. Z. M. Islam, D. Das, "Introduction of PMI-SO integrated with predictive and lexicon based features to detect cyberbullying in bangla text using machine learning," in *Proceedings of 2nd International Conference on Artificial Intelligence: Advances and Applications*, 2022, pp. 685–697, doi: 10.1007/978-981-16-6332-1_56.

[4] M. T. Ahmed, M. Rahman, S. Nur, A. Islam, and D. Das, "Deployment of machine learning and deep learning algorithms in detecting cyberbullying in Bangla and romanized bangla text: A comparative study," in *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Feb. 2021, pp. 1–10, doi: 10.1109/ICAECT49130.2021.9392608.

[5] A. K. Das, A. A. Asif, A. Paul, and M. N. Hossain, "Bangla hate speech detection on social media using attention-based recurrent neural network," *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 578–591, Apr. 2021, doi: 10.1515/jisys-2020-0060.

[6] N. Romim, M. Ahmed, H. Talukder, and M. S. Islam, "Hate speech detection in the bengali language: A dataset and its baseline evaluation," in *Proceedings of International Joint Conference on Advances in Computational Intelligence*, 2021, pp. 457–468, doi: 10.1007/978-981-16-0586-4_37.

[7] T. Islam, N. Ahmed, and S. Latif, "An evolutionary approach to comparative analysis of detecting Bangla abusive text," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2163–2169, Aug. 2021, doi: 10.11591/eei.v10i4.3107.

[8] R. Kumar, B. Lahiri, and A. K. Ojha, "Aggressive and offensive language identification in Hindi, Bangla, and English: A comparative study," *SN Computer Science*, vol. 2, no. 1, pp. 1–20, Feb. 2021, doi: 10.1007/s42979-020-00414-6.

[9] T. Mandl *et al.*, "Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in Indo-European languages," in *Proceedings of the 11th forum for information retrieval evaluation*, Dec. 2019, doi: 10.1145/3368567.3368584.

[10] S. Bhattacharya *et al.*, "Developing a multilingual annotated corpus of misogyny and aggression," *arXiv preprint arXiv:2003.07428*, 2020, doi: 10.48550/arXiv.2003.07428.

[11] P. Chakraborty and M. H. Seddiqui, "Threat and abusive language detection on social media in Bengali language," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, May 2019, pp. 1–6, doi: 10.1109/ICASERT.2019.8934609.

[12] M. H. Seddiqui, A. A. M. Maruf, and A. N. Chy, “Recursive suffix stripping to augment Bangla stemmer,” in *International Conference Advanced Information and Communication Technology (ICAICT)*, 2016.

[13] M. Rahman and M. Seddiqui, “Comparison of classical machine learning approaches on bangla textual emotion analysis,” *arXiv preprint arXiv:1907.07826*, 2019, doi: 10.48550/arXiv.1907.07826.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018, doi: 10.48550/arXiv.1810.04805.

[15] S. Ahammed, M. Rahman, M. H. Niloy, and S. M. M. H. Chowdhury, “Implementation of machine learning to detect hate speech in Bangla language,” in *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, Nov. 2019, pp. 317–320, doi: 10.1109/SMART46866.2019.9117214.

[16] M. Jahan, I. Ahamed, M. R. Bishwas, and S. Shatabda, “Abusive comments detection in Bangla-English code-mixed and transliterated text,” in *2019 2nd International Conference on Innovation in Engineering and Technology (ICIET)*, Dec. 2019, pp. 1–6, doi: 10.1109/ICIET48527.2019.9290630.

[17] O. Sharif and M. M. Hoque, “Automatic detection of suspicious bangla text using logistic regression,” in *Intelligent Computing and Optimization: Proceedings of the 2nd International Conference on Intelligent Computing and Optimization 2019 (ICO 2019)*, 2020, pp. 581–590, doi: 10.1007/978-3-030-33585-4_57.

[18] E. A. Emon, S. Rahman, J. Banarjee, A. K. Das, and T. Mittra, “A deep learning approach to detect abusive Bengali text,” in *2019 7th International Conference on Smart Computing and Communications (ICSCC)*, Jun. 2019, pp. 1–5, doi: 10.1109/ICSCC.2019.8843606.

[19] M. M. Rayhan, T. A. Musabe, and M. A. Islam, “Multilabel emotion detection from bangla text using BiGRU and CNN-BiLSTM,” in *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, Dec. 2020, pp. 1–6, doi: 10.1109/ICCIT51783.2020.9392690.

[20] M. S. Jahan, M. Haque, N. Arhab, and M. Oussalah, “Banglahatebert: Bert for abusive language detection in Bengali,” in *Proceedings of the Second International Workshop on Resources and Techniques for User Information in Abusive Language Analysis*, Jun. 2022, pp. 8–15.

[21] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[22] S. Xu, Y. Li, and Z. Wang, “Bayesian multinomial Naïve Bayes classifier to text classification,” in *Advanced Multimedia and Ubiquitous Engineering*, Singapore: Springer, 2017, pp. 347–352, doi: 10.1007/978-981-10-5041-1_57.

[23] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, “Random forests and decision trees,” *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, pp. 272-278, Sep. 2012.

[24] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[25] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997, doi: 10.1109/78.650093.

[26] M. R. Karim, B. R. Chakravarthi, J. P. McCrae, and M. Cochez, “Classification benchmarks for under-resourced Bengali language based on multichannel convolutional-LSTM network,” in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2020, pp. 390–399, doi: 10.1109/DSAA49011.2020.00053.

[27] T.-T. Wong and P.-Y. Yeh, “Reliable accuracy estimates from k-fold cross validation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1586–1594, Aug. 2019, doi: 10.1109/TKDE.2019.2912815.

[28] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” *arXiv preprint arXiv:1607.01759*, 2016, doi: 10.48550/arXiv.1607.01759.

[29] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, “Learning word vectors for 157 languages,” *arXiv preprint arXiv:1802.06893*, 2018, doi: 10.48550/arXiv.1802.06893.

BIOGRAPHIES OF AUTHORS

Md. Nesarul Hoque is an assistant professor at the Bangabandhu Sheikh Mujibur Rahman Science and Technology University in Bangladesh. He works in the Department of Computer Science and Engineering. He received Bachelor of Science and Master of Science in Engineering degrees from the University of Chittagong, Bangladesh. At present, he is pursuing a Doctor of Engineering degree from the University of Chittagong, Bangladesh. He has some articles at renowned international conferences. His research interests cover the areas of machine learning, deep learning, natural language processing, and information retrieval. He can be contacted at email: mnhshisir@gmail.com.

Md. Hanif Seddiqui has completed his Master of Engineering and Doctor of Engineering degrees from Toyohashi University of Technology, Japan with a number of research awards. He has also completed his post-doctoral fellowship under the European Erasmus Mundus cLINK Programme in the DISP Laboratory, University Lumiere Lyon 2, France, and a short fellowship from KDE Lab, Toyohashi University of Technology, Japan. Currently, he is taking responsibility as a professor in the Department of Computer Science and Engineering, University of Chittagong, Bangladesh. He has a few more remarkable contributions on instance matching, Semantic NLP, healthcare and geospatial domain using semantic technology as well. Dr. Seddiqui is a regular member of the conference program committees in the areas of machine learning, data mining, and semantic web as well as a reviewer of some journals. He can be contacted at email: hanif@cu.ac.bd.